What is claimed is:

1. A method comprising:

providing a common source code unit including production source code and test source code, the test source code having test methods for testing the production source code;

producing executable production code from the production source code and executable test code from the test source code; and

providing a global switch specifying whether to load the executable test code with the executable production code into a production environment.

2. The method of claim 1 further comprising:

loading the executable production code in the production environment without the executable test code in response to a setting of the global switch.

3. The method of claim 1 further comprising:

executing the executable production code in a development environment; and

loading the executable test code with the executable production code in the development environment in response to a setting of the global switch.

4. The method of claim 1 further comprising:

executing the executable production code in the production environment;

loading the executable production code in the production environment without the executable test code in response to a setting of the global switch;

changing the setting to specify that the executable test code is to be loaded with the executable production code in the production environment; and

loading the executable test code with the executable production code in the production environment, in response to the changed setting.

5. The method of claim 1 wherein the production source code and the test source code are generated using at least one of a procedural programming language including one of C, Fortran and Pascal, an object oriented programming language including at least one of a

advanced business application program language (ABAP), Java programming language, C++ programming language and C# programming language.

6. The method of claim 1 further comprising checking static references from the production source code to the test source code.

7. The method of claim 1 further comprising checking dynamic references from the executable production code to the executable test code.

8. The method of claim 1 wherein the executable production code and executable test source code is produced using a compiler.

9. The method of claim 1 wherein the test source code has access to a functionality of the production source code.

10. The method of claim 1 further comprising synchronizing changes to the test source code and the production source code.

11. The method of claim 1 wherein the production source code and test source code are implemented in a unit test environment.

12. The method of claim 1 wherein the common source unit includes production source code and test source code sharing a same compilation unit.

13. The method of claim 1 wherein the common compilation unit includes executable production code and executable test code sharing a same compilation unit.

14. The method of claim 1 wherein the test method includes test assertion methods providing instructions for verifying an expected state of production source code.

15. A computer system comprising:

a common source unit having production source code and test source code with test methods for testing the production source code; and

a means for producing a common compilation unit having executable production code based on the production source code and executable test code based on the test class source code or only the executable production code in response to a value of a system global switch.

16 . The system of claim 15 further comprising a means for:

executing the executable production code in the production environment; and

loading the executable production code in the production environment, without loading the executable test code, in response to the value of the global switch.

17. The system of claim 15 further comprising a means for:

executing the executable production code in a development environment; and

loading the executable test code with the executable production code in the development environment, in response to the value of the global switch.

18. The system of claim 15 further comprising a means for:

executing the executable production code in the production environment;

loading the executable production code in the production environment, without loading the executable test code, in response to the value of the global switch;

changing the value of the global switch to specify that the executable test code is to be loaded with the executable production code in the production environment; and

loading the executable test code with the executable production code in the production environment, in response to the changed value of the global switch.

19. The system of claim 15 wherein the production source code and/or the test source code are generated using at least one of a procedural programming language including one of C, Fortran and Pascal, an object oriented programming language including at least one of a advanced business application program language (ABAP), Java programming language, and C++ programming language and C# programming language.

20. The system of claim 15 further comprising a means for checking static references from the production source code to the test source code.

21. The system of claim 15 further comprising a means for checking dynamic references from the executable production code to the executable test code.

22. The system of claim 15 wherein the executable production code and/or executable test source code is produced using a compiler.

23. The system of claim 15 wherein the test source code has access to a functionality of the production source code.

24. The system of claim 15 further comprising a means for synchronizing changes to the test source code and the production source code.

25. The system of claim 15 wherein the production source code and test source code are implemented in a unit test environment.

26. The system of claim 15 wherein the common source unit includes production source code and test source code share a same compilation unit.

27. The system of claim 15 wherein the common compilation unit includes executable production code and executable test code sharing a same compilation unit.

28. The system of claim 15 wherein the test method includes test assertion methods providing instructions for verifying an expected state of production source code.

29. A method comprising the steps of:

a step of providing a common source unit having production source code and test source code, the test source code having test methods for testing the production source code;

a step of producing executable production code based on the production source code and executable test code based on the test source code; and

a step of providing a global switch specifying whether to load the executable test code with the executable production code in a production environment.

30. The method of claim 29 further comprising the steps of:

a step of executing the executable production code in the production environment; and

a step of loading the executable production code in the production environment, without loading the executable test code in response to the global switch.

31. The method of claim 29 further comprising the steps of:

a step of executing the executable production code in a development environment; and

a step of loading the executable test code with the executable production code in the development environment, in response to the global switch.

32. The method of claim 29 further comprising the steps of:

a step of executing the executable production code in the production environment;

a step of loading the executable production code in the production environment, without loading the executable test code, in response to the global switch;

a step of changing the global switch to specify that the executable test code is to be loaded with the executable production code in the production environment; and

a step of loading the executable test code with the executable production code in the production environment, in response to the changed global switch.

33. The method of claim 29 further comprising the step of checking static references from the production source code to the test source code.

34. The method of claim 29 further comprising the step of checking dynamic references from the executable production code to the executable test code.

35. The method of claim 29 further comprising the step of synchronizing changes to the test source code and the production source code.